APPLICANTS' REMARKS IN SUPPORT OF THE PRE-APPEAL BRIEF REQUEST FOR REVIEW

The Examiner rejected claims 1-16 under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,345,272 to Witkowski *et al.* ("Witkowski"). As set forth in detail below, Applicants submit that the Examiner's basis for the § 102(e) rejection is improper because Witkowski fails to teach each and every element of the claims of the present invention. Specifically, Witkowski does not teach performing a SQL join operation, a feature recited in the each of the three independent claims of the present application.

The Rejections of Claims 1-16 under 35 U.S.C. § 102(e)

In order for a reference to anticipate under § 102(e), the Examiner must show that the reference teaches each and every feature recited by the claims of the present application. *See* MPEP § 2131. In this regard, Applicants submit that the Examiner's rejection lacks merit. That is, the rejection of independent claims 1, 9, and 16 are improper because at least one element of each of these claims is not taught by the cited reference.

One embodiment of the present invention, for example as recited by claim 1, comprises a query generator for generating a query for obtaining selected data from a database. The database has a number of detail tables in which data is stored, and the query generator comprises a processor that is coupled to the database in use. The processor is adapted to receive an input indicating the selected data to be obtained to generate a first query. The input is then analyzed to determine whether the input requires a joining of data in a plurality of different detail tables, and an aggregation step. If it is determined that the input requires a joining of data in a plurality of different detail tables, the processor modifies the content of the first query indicating the selected data to be obtained to generate a second query. The second query is adapted to cause the database

to generate a query that aggregates data within each of a plurality of detail tables. Then, a SQL join operation is performed to join the aggregated data from each of the plurality of detail tables.

In contrast, Witkowski discloses a method and apparatus for rewriting aggregate queries to access a materialized view under certain predetermined conditions. *See* Abstract. It is often desirable to access information from a database, which may store data in data containers, referred to as tables. *See* Col. 1, lines 11-20. For various reasons, it may not be desirable to allow a user to access all of the columns of a table. *See* Col. 1, lines 28-37. Instead, a user may be allowed to indirectly access the appropriate columns of a table through a "view." *Id.* A drawback of a view, however, is that it presents data that is extracted or derived from existing tables. *See* Col. 1, lines 38-44. Since the data presented by a conventional view is gathered and derived in response to a specific query, it is not stored after the query accessing the view has been processed. *See* Col. 1, lines 54-63. This method of processing views can become time and cost prohibitive. *Id.*

To overcome this disadvantage, skilled artisans use materialized views. *See* Col. 1, line 64 - Col. 2, line 6. A materialized view is a view for which a copy of the view data is stored separately from the existing tables from which the data was originally gathered and derived. *Id.* An advantage of a materialized view is that they eliminate the overhead associated with gathering and deriving the view data every time a query accesses the view. *Id.*

FIG. 2 of Witkowski graphically illustrates the ability to rewrite a query such that it accesses a materialized view. *See* Col. 3, line 66 - Col. 4, line 6. In FIG. 2, an aggregate query 210 requests summary information from sales table 250 using the SQL command "SUM $AMT." *See* Col. 4, lines 7-16. The figure also shows a query that defines a materialized view 270. *See* Col. 4, lines 17-18. The material view query uses the *same command* "SUM $AMT" as in the aggregate query 210, but represents it using a different name, *e.g.*, "Sum_Sales." *See* FIG. 2. Thus, FIG. 2

illustrates the two types of queries described above: (i) a query that requests information from a existing table in a database; and (ii) a query that generates a materialized view. In addition, FIG. 2 shows a third query, 280, which represents query 210 *rewritten* to access the materialized view *instead of* the original data from the database. *See* Col. 4, lines 24-30. Again, query 210 is *rewritten* as query 280. Therefore, query 280 does not join two sets of aggregated data. Rather, there is only a single aggregation defined by "SUM $AMT." "SUM $AMT" is *the same as* "SUM_Sales," however the latter is the representation of "SUM $AMT" in the materialized view. Consequently, query 280 is not joining any aggregated sets of data. Rather, it is merely a rewritten query that accesses the materialized view instead of the existing data in the database.

Another way to explain Witkowski is as follows. Witkowski explicitly states that the purpose of the invention disclosed therein is to provide a method and apparatus for rewriting aggregate queries to access a materialized view. *See* abstract. To accomplish this, Witkowski generates three separate queries: (i) an aggregate query 210; (ii) a materialized view query 270; and (iii) a query 280 that accesses the materialized view. Each of these three types of queries is discussed below.

The aggregate query 210 requests summary information from a first table, *e.g.,* sales table 250. *See* Col. 4, lines 7-16. This query aggregates the data in the sales table 250, which contains many different columns of data. *Id.* Witkowski also discloses a materialized view query 270, that aggregates data from the "$AMT" column of the sales table 250. The result of the materialized view query 270 is a second table MVSales, that includes a column titled "Sum_Sales." Thus, Witkowski discloses two tables, *e.g.,* a sales table 250 and a MVSales table.

Witkowski teaches that it is often desirable to access a materialized view table, *e.g.,* MVSales, rather than a table with several columns, *e.g.,* sales table 250, for various reasons. *See*

Col. 1, lines 11-20. In order to do this, Witkowski teaches that aggregate query 210 may be rewritten to access the materialized view table MVSales instead of the sales table 250. *See* Col. 3, line 66 - Col. 4, line 47. This is performed by rewriting aggregate query 210 as query 280. *Id.*

By rewriting the query, aggregate query 280 performs an aggregation of the "Sum_Sales" column of the MVSales table only. In this manner, the first sales table 250 is not referenced by query 280. Rather, query 280 accesses only the second table MVSales. Indeed, query 280, illustrated in FIG. 2, only references the "Sum_Sales" column of the MVSales table created by materialized view query 270. *See* FIG. 2. Applicants argument is further supported by the fact that there is no reference to the sales table 250 in query 280.

It is important to note that the three queries disclosed by Witkowski are separate and distinct. In other words, query 210 is one way of creating a table by accessing the first sales table 250. Query 270 is a separate query that creates a materialized view by accessing the first sales table 250. In contrast to these two queries, query 280 does not access first sales table 250. Instead, it is a separate query that aggregates data only from a second table created by materialized view query 270.

In the previous Office Actions, the Examiner repeatedly contends that query 280 joins the data from table 250 and the summary table MVSales. The Examiner provides no support for this contention, however, and the argument lacks basis. It appears that the Examiner's contention stems from the fact that the "Sum_Sales" column is a result of the aggregation of data from table 250. However, the mere fact that the "Sum_Sales" column is a result of aggregation of data from table 250 does not mean that query 280 implements a join of the two tables. Rather, query 270 aggregates the data from table 250, at which point it is stored as a separate table. *See* Col. 1, line 64 - Col. 2, line 6. When desired, query 280 aggregates the data from the "Sum_Sales" column (recall

this is stored separately) without any reference to, or consideration of, table 250. *See* FIG. 2. Thus, query 280 only accesses one table, and thus cannot logically implement a join. *Id.*

Another way of explaining the error in the Examiner's interpretation is as follows. Both Applicants and the Examiner can agree that sales table 250 is a separate table. The manner in which sales table 250 is formed, *e.g.*, the specific query, is of no consequence. Moreover, query 210 aggregates data from sales table 250, and a skilled artisan would not assert that query 210 performs a join operation. The function of query 280 is tantamount to the function of query 210. That is, the MVSales table is stored separately in a memory. *See* Col. 1, line 64 - Col. 2, line 6. Since it is a separate table, the manner in which MVSales is formed is of no consequence. Further, query 280 simply aggregates a row, *e.g.*, "Sum_Sales," from the MVSales table. Thus, a skilled artisan would recognize that query 280 does not perform a join operation.

At least because Witkowski fails to teach "perform[ing] a SQL join operation to join the aggregated data from each of the plurality of detail tables," as recited by the independent claims, Applicants submit that the Examiner's § 102 rejection is improper, and must be withdrawn. In light of the deficiencies in the Examiner's anticipation rejection under 35 U.S.C. § 102(e), Applicants respectfully request reconsideration and issuance of a Notice of Allowance for the entirety of the pending claims.

Respectfully Submitted,

Dated: November 20, 2007

By: _____

Siddhesh V. Pandit, Reg. No. 58,572
Bingham McCutchen LLP
2020 K Street, NW
Washington, DC 20006
(202) 373-6513 Telephone
(202) 373-6001 Facsimile